

# **EXHIBIT A:**

## **Redacted Copy of Invention Disclosure Form**

**(4 pages)**

***Response to Non-Final Office Action filed October 22, 2007***

→ **From:** Ron Karr  
**Sent:** Thursday, October 16, 2003 9:25 PM  
**To:** LIST-Inventions  
**Cc:** Ron Karr  
**Subject:** VRTS 0623: Coherently sharing any form of ?instant? snapshot separately from base volumes

## VERITAS CONFIDENTIAL

---

### Invention Disclosure Form

---

**Name:** Ronald Karr  
**Telephone:** 650-527-8439  
**Email:** tron@veritas.com

2. Provide a brief descriptive title of the invention:  
**Coherently sharing any form of "Instant" snapshot separately from base volumes**

3. In 50 words or less, please provide an abstract or summary of the invention:  
**SANVM as well as CVM and VxVM support instant snapshots, as described in various other patents. This patent uses SANVM, CVM, XVM, or perhaps Switch-based VM with LUN tunneling to provide access to an instant snapshot from a second system.**

4. List all inventors who contributed to this invention (if more than four, please provide the additional names in the space provided):

**1st Contributor Name:** Ronald Karr  
**1st Contributor Division:** Engineering  
**1st Contributor Campus:** MTV  
**1st Contributor Phone:** 650-527-8439  
**1st Contributor eMail:** tron@veritas.com  
**VERITAS Employee?** yes

**2nd Contributor Name:** Anand Kekre  
**2nd Contributor Division:** Engineering  
**2nd Contributor Campus:** Pune  
**2nd Contributor Phone:**  
**2nd Contributor eMail:** anand.kekre@veritas.com  
**VERITAS Employee?** yes

**3rd Contributor Name:**  
**3rd Contributor Division:**  
**3rd Contributor Campus:**  
**3rd Contributor Phone:**  
**3rd Contributor eMail:**  
**VERITAS Employee?**

**4th Contributor Name:**  
**4th Contributor Division:**  
**4th Contributor Campus:**  
**4th Contributor Phone:**  
**4th Contributor eMail:**  
**VERITAS Employee?**

## **Introduction**

SANVM, XVM, and some of the designs for switch-based VxVM depend on extending the architecture of VERITAS Volume Manager, by giving it a very flexible form of storage sharing. They are all based on *distributed block virtualization*, which extends the basic concept of single node and symmetrically clustered virtual block devices into flexibly distributed block virtualization components.

SANVM, XVM, and switch-based VxVM differ in how they intend to distribute components of block virtualization, but they share many structural similarities. Distributed virtualization itself is an important concept that will likely be very important to the future of block storage.

## **Basic software structure**

Distributed block virtualization basically distributes a description of how a virtual block device (for example, a logical volume or a virtual LUN) relates to underlying storage, as well as how distributed block virtualization components might relate in order to accomplish certain functions.

Block virtualization components can live in various places. They can be software layers on servers that run file systems or databases. They can be in the disk arrays. They can be in intermediate devices residing in storage networks between hosts and disk arrays. They can be in the host bus adapters that connect a host to a network. Any of these components could participate in some form of distributed virtualization. It is also possible that any of these components could implement a non-distributed virtualization even though other components do implement distributed virtualization. Components can cooperate to provide distributed virtualization either across a layer or between layers, or layered components can implement isolated virtualization layers where a higher layer uses a lower layer in the same way the upper layer would simply use a disk drive.

A typical implementation of distributed block virtualization will have components running on some collection of nodes to identify and organize some collection of underlying logical disks (*device management*). Additionally, they will have *configuration management* components that store and retrieve information that describes the logical structure associated with the virtualization (how logical and physical disks are divided, aggregated, mirrored, and so on). Various *distributed virtualization coordination* components analyze the logical structure and communicate information about that structure to the various *I/O engines* that receive and transform read and write requests (the incoming I/O stream) to virtual block devices and convert them into network messages and reads and writes to underlying logical and physical disk drives.

The configuration associated with a virtual block device may change over time, such as to add or remove mirrors; migrate data to new storage; increase or decrease the size of the device; create, manipulate, or remove snapshots; add structure for a new capability; etc. Changes that affect several I/O engines must often result in coherent updates that are effectively *atomic* relative to some aspect of the I/O stream. This is generally done through some kind of distributed transactional update, coupled with some form of *I/O quiesce*, which temporarily blocks activity while distributed structures are updated.

Figure 1 graphically depicts a typical logical volume-based block virtualizer, such as that implemented by SANVM. A coordinator, in SANVM called a *volume server*, serves the configuration management and distributed virtualization coordination functions. The SANVM volume server reads and updates configuration information from VxVM configuration databases (which are actually stored on the disks that are being virtualized) in persist and retrieve the logical configuration. The logical configuration for a single volume is depicted in figure 1 as a tree of virtual objects. This tree is coherently communicated and updated to all the nodes that share access to that volume. Each node can then independently use that tree of virtual objects to transform the I/O stream coming into the I/O engine on that node into I/O requests directly to the physical disks that underlie the volume.

**SANVM also has another software component (called a *log coordinator*) that is an example of a special component that coordinates some virtualization function. The log coordinator in SANVM coordinates update activity associated with recoverable mirroring and snapshots. Such *function coordinator* components are likely to be prevalent within many distributed virtualization implementations.**

12. Has the invention been built or implemented? If so, please provide the particulars of the first time it was successfully built or implemented (when, where, by whom, and evidence of this event including written or on-line pointers to documentary evidence):

→ **It is in CVM 4.0 and will be in SANVM.**

15. Please provide the names of products that the invention is, or will be used in (if any):

Primary product invention is used in: **SANVM**

Other products invention may be used in: **CVM and Switch versions of SANVM**

